5. Hiromi

Program Name: Hiromi.java **Input File: hiromi.dat**

Sudoku is a popular, logic-based game that anyone, who can count from 1-9, can play! A valid sudoku puzzle is a 9 by 9 grid of numbers such that each row, each column, and each 3x3 sub-grid region has the numbers 1-9 with no missing numbers or no duplicate numbers. A puzzle is traditionally given incomplete, and it is up to the player to fill in all the missing spaces. As the below puzzle shows, each row (horizontal), each column (vertical), and each of the nine 3x3 sub-grids, all have the numbers 1-9. See the below puzzle for a valid Sudoku puzzle, as well as the layout of the 9 sub-grids. (The darker, bolder lines denote the sub-grids.)

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Hiromi loves solving Sudoku's but is known to make a mistake time-to-time. In an effort to ensure the final attempts are correct, Hiromi would like your help writing source code that verifies if a given attempt at a Sudoku puzzle is in fact correct, or not. Can you help with this?

Input: Input will begin with an integer N, the number of test cases. N is in the range [1,10]. Each of the following N test cases will start with 17 dashes to serve as a divider between each of the puzzles. Following the dashes will be the 9 x 9 Sudoku puzzle. All, individual numbers will be separated by a space to simplify the reading of the input. Each number present is guaranteed to be in range of [1,9].

Output: For each test case, you are to output: "`Puzzle #X: true`" if the puzzle meets the described criteria for a valid Sudoku puzzle, or "`Puzzle #X: false`", where `X` is the current test case. Remember, all nine rows, all nine columns, and all 9 sub-grids must have the numbers 1-9, to be a valid puzzle.

Sample input and output on next page…

~ Hiromi continued… ~

Sample input:

```
5
-----------------
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
-----------------
1 2 3 4 5 6 7 8 9
4 5 6 7 8 9 1 2 3
7 8 9 1 2 3 4 5 6
2 3 4 5 6 7 8 9 1
5 6 7 8 9 1 2 3 4
8 9 1 2 3 4 5 6 7
3 4 5 6 7 8 9 1 2
6 7 8 9 1 2 3 4 5
9 1 2 3 4 5 6 7 8
-----------------
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 7
-----------------
8 3 5 4 1 6 9 2 7
2 9 6 8 5 7 4 3 1
4 1 7 2 9 3 6 5 8
5 6 9 1 3 4 7 8 2
1 2 3 6 7 8 5 4 9
7 4 8 5 2 9 1 6 3
6 5 2 7 8 1 3 9 4
9 8 1 3 4 5 2 7 6
3 7 4 9 6 2 8 1 5
-----------------
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 3
```

Sample output:

```
Puzzle #1: true
Puzzle #2: true
Puzzle #3: false
Puzzle #4: true
Puzzle #5: false
```